

Še o branju datotek

Datoteke beremo že dolgo, od drugega tedna. Zamolčal pa sem vam, oni dan, da imajo datoteke tudi metode. Preprosto zato, ker metod še nismo srečali in tudi nobene potrebe ni bilo, da bi jih ravno ob datotekah. Zdaj jih lahko. So tri, štiri, nezahtevne in, no, tudi ne nujno zelo uporabne. (V čemer se je skrival dodatni razlog, da jih nisem omenjal. Študenti imajo sicer navado, da kličejo te metode takrat, ko jih ne potrebujejo, saj bi zadoščala že navadna zanka `for`. Zato sem vas v svoji didaktični zvitosti raje prisilil, da ste se jih temeljito navadili brati z zanko `for` in bo skušnjava po klicanju nepotrebnih metod, upam, manjša.)

Datoteke smo vedno brali z zanko `for` (dokler nismo spoznali `csv.reader` in `csv.DictReader`, pa še kaj podobnega bomo).

```
for vrstica in open("kolesa-z-glavo.txt"):
    print(vrstica)
```

```
kolo,razdalja,višina,lastnik,leto nakupa
```

```
Cube,5031,159,Janez,2017
```

```
Stevens,3819,1284,Ana,2012
```

```
Focus,3823,1921,Benjamin,2019
```

Datoteka ima, kot napisano, metode.

```
f = open("kolesa-z-glavo.txt")
```

Ena je `read()`; ta prebere celo datoteko v niz.

```
f.read()
```

```
'kolo,razdalja,višina,lastnik,leto nakupa\nCube,5031,159,Janez,2017\nStevens,3819,1284,Ana,2012\n'
```

To stori le enkrat. Zdaj je datoteka "potrošena".

```
f.read()
```

```
''
```

Če želimo z njo početi še kaj, jo bomo morali odpreti ponovno. (Obstaja način, da jo "prevrtimo" nazaj, vendar tega ne bomo počeli.)

```
f = open("kolesa-z-glavo.txt")
```

Druga je `readlines()`; ta prebere vse vrstice datoteke v seznam nizov.

```
f = open("kolesa-z-glavo.txt")
```

```
f.readlines()
```

```
['kolo,razdalja,višina,lastnik,leto nakupa\n',
 'Cube,5031,159,Janez,2017\n',
 'Stevens,3819,1284,Ana,2012\n',
```

```
'Focus,3823,1921,Benjamin,2019']
```

Tretja je `readline()`. Ta prebere eno vrstico.

```
f = open("kolesa-z-glavo.txt")
f.readline()
```

```
'kolo,razdalja,višina,lastnik,leto nakupa\n'
```

Vsakič naslednjo.

```
f.readline()
```

```
'Cube,5031,159,Janez,2017\n'
```

```
f.readline()
```

```
'Stevens,3819,1284,Ana,2012\n'
```

Kar je prebrano, je prebrano in se ne vrne.

Izmed teh treh boste največkrat potrebovali zadnjo (`readline()`), včasih tudi prvo (`read()`). Če vas zanima uporabiti `readlines()`, si boste navadno naredili uslugo, če boste raje naredili zanko čez datoteko.

`readline()` bomo uporabili, kadar datoteka nima "homogenega" formata: različne vrstice imajo, recimo, različen pomen in praktično nam jih je brati eno za drugo.

"Kar je prebrano, je prebrano," sem napisal z mislijo na tale primer:

```
f = open("kolesa-z-glavo.txt")

stolpci = f.readline().split(",")
for vrstica in f:
    print(vrstica)
```

```
Cube,5031,159,Janez,2017
```

```
Stevens,3819,1284,Ana,2012
```

```
Focus,3823,1921,Benjamin,2019
```

Prvo vrstico smo prebrali v `stolpci`; morda nam pride prav, morda smo se je hoteli le znebiti. Ostale vrstice beremo kar s `for`. Prve nam ne bo prebrala, ker je že prebrana.

Četrta omembe vredna metoda, je `close`.

```
f = open("kolesa-z-glavo.txt")
```

```
f.readline()
```

```
'kolo,razdalja,višina,lastnik,leto nakupa\n'
```

```

f.readline()
'Cube,5031,159,Janez,2017\n'
f.close()
f.readline()

-----
ValueError                                Traceback (most recent call last)
Cell In[14], line 1
----> 1 f.readline()

```

ValueError: I/O operation on closed file.

Ko datoteko zapremo je ne moremo več brati. :)

Zakaj bi kdo zaprl datoteko? Naj ostane odprta! Kot prvo: program na MS Windows ima lahko samo 512 hkrati odprtih datotek. Za druge operacijske sisteme ne vem. Svojčas je bilo omejeno tudi število hkrati odprtih datotek v vseh programih skupaj ... OK, nam je mar? Ne res.

Python sam zapira datoteke.

```

f = open("kolesa-z-glavo.txt")
f.readline()

```

```

f = open("kolesa.txt")

```

Ker se `f` zdaj nanaša na neko drugo datoteko in iz prve tako ali tako ne moremo več brati, jo Python zapre.

Še bolj pogosta je situacija, ko datoteko odpremo znotraj funkcije. Ko je funkcije konec, ime, ki se je nanašalo nanjo (recimo `f`) izgine, do datoteke ne moremo več in Python jo zapre. Skratka: `close` ne boste klicali prav pogosto.

Pisanje datotek

Naučivši se oblikovati nize smo izpisali zelo lepo tabelico.

```

voznje = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}
razdalje = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}
visine = {"Canyon": 0, "Cube": 0, "Nakamura": 0, "Stevens": 0}

for vrstica in open("kolesa.txt"):
    kolo, razdalja, visina = vrstica.split(",")
    voznje[kolo] += 1
    razdalje[kolo] += int(razdalja)
    visine[kolo] += int(visina)

sk_voz = sum(voznje.values())

```

```

sk_raz = sum(razdalje.values())
sk_vis = sum(visine.values())

print()
print(f"{'Kolo':6}{ 'Število voženj':>19}{ 'Razdalja':>19}{ 'Višina':>19}")
print("-" * (6 + 3 * 19))
for kolo in voznje:
    voz, raz, vis = voznje[kolo], razdalje[kolo], visine[kolo]
    print(f"{'kolo:12'}{'voz:5'} ({voz / sk_voz:>5.1%}) {'raz:10'} ({raz / sk_raz:>5.1%}) {'vis:10'}")
print("-" * (6 + 3 * 19))

```

Kolo	Število voženj	Razdalja	Višina
Canyon	26 (26.0%)	2766 (39.6%)	26392 (27.0%)
Cube	43 (43.0%)	3174 (45.4%)	66705 (68.3%)
Nakamura	22 (22.0%)	439 (6.3%)	1119 (1.1%)
Stevens	9 (9.0%)	607 (8.7%)	3395 (3.5%)

Tako je lepa, da bi jo lahko zapisali kar v datoteko. Datoteka naj se imenuje tabelica-kolesa.txt.

Poskusimo.

```
f = open("tabelica-kolesa.txt")
```

```

-----
FileNotFoundError                                Traceback (most recent call last)
Cell In[17], line 1
----> 1 f = open("tabelica-kolesa.txt")

```

```

File ~/opt/miniconda3/envs/prog/lib/python3.11/site-packages/IPython/core/interactiveshell.p
279 if file in {0, 1, 2}:
280     raise ValueError(
281         f"IPython won't let you open fd={file} by default "
282         "as it is likely to crash IPython. If you know what you are doing, "
283         "you can use builtins' open."
284     )
--> 286 return io_open(file, *args, **kwargs)

```

```
FileNotFoundError: [Errno 2] No such file or directory: 'tabelica-kolesa.txt'
```

Seveda ne. Datoteke ni; ni česa odpreti. Seveda ne: datoteko bi radi naredili.

Funkciji `open` moramo dati še en argument - način, na katerega želimo odpreti datoteko. Ta je lahko `"r"` (branje), `"w"` (pisanje) ali `"a"` (dodajanje v obstoječo datoteko, kot *append*). (Dodali bi lahko še eno črko `t`, ki bi povedala, da gre za besedilno in ne binarno datoteko, b. Pri tem predmetu nas to ne zanima.)

```
f = open("tabelica-kolesa.txt", "w")
```

Ker je `f` datoteka, ima običajne metode datotek.

```
f.readline()
```

```
-----
UnsupportedOperation                                Traceback (most recent call last)
Cell In[19], line 1
----> 1 f.readline()
```

UnsupportedOperation: not readable

Metodo ima, to že, le poklicati je ne smemo. V to datoteko pišemo. Nič, kar se začne z `read`, ne bo delovalo. Pač pa zanjo deluje `write` (ki ga prej, ko smo bili še pri branju, nismo niti pokazali, ker - uganili ste - ne bi delal, tako kot tu ne dela branje).

```
f.write(f"{'Kolo':6}{ 'Številó voženj':>19}{ 'Razdalja':>19}{ 'Višina':>19}\n")
f.write("-" * (6 + 3 * 19) + "\n")
for kolo in voznje:
    voz, raz, vis = voznje[kolo], razdalje[kolo], visine[kolo]
    f.write(f"{kolo:12}{voz:5} ({voz / sk_voz:>5.1%}) {raz:10} ({raz / sk_raz:>5.1%}) {vis:10}\n")
f.write("-" * (6 + 3 * 19) + "\n")
```

64

Vse je enako kot prej, le

- `print` zamenjamo s `f.write`,
- na konec vsake vrstice dodamo `\n`.

`print` je šel v novo vrstico sam, `write` ne bo.

Razlik med `print` in `write` je par:

- `write` ne gre sam v novo vrstico,
- `print` sprejme več argumentov in lahko so poljubnih tipov, od nizov do števil in seznamov in množic in česarkoli. `write` sprejme en sam argument in ta mora biti niz.
- `print` ne vrne rezultata (vrne `None`), `write` vrne število bajtov, zapisanih v tem klicu. Zakaj? Pojma nimam. Ampak to je tista 64, ki se je izpisala na koncu: pomeni $6 + 3 * 19 = 63$ minusov in en `\n`.

Če zdaj odpremo datoteko `tabelle-kolesa.txt`, vidimo, da je ... popolnoma prazna. Kam so potem šli vsi ti `write`-i? Vključno z zadnjimi 64?

Računalnik v datoteke na zapisuje vsega sproti, temveč šele, ko se nabere. Sicer bi moral stalno, za vsako vrstico posebej dostopati do diska; celo v času SSD diskov bi bilo to počasno. In tu ... se pač še ni nabralo. Prav, ampak zdaj smo končali. Kako Pythonu povedati, da je končano? Preprosto: zapremo datoteko. :)

```
f.close()
```

(Obstaja še en način: `f.flush()` zapiše, kar se je nabralo doslej, datoteke pa ne zapre.)

To pomeni, da bomo pri pisanju vedno poklicali `close()`, da bomo zapisali vsebino datoteke? Ne. Jaz ga navadno ne kličem. Zato smo se pa ob branju pogovarjali o tem, kdaj Python zapre datoteko. Če jo pišemo znotraj funkcije, bo na koncu funkcije, ko lokalna spremenljivka (recimo `f`) izgine, tudi datoteka sama zaprta.

Pravilno odpiranje datotek

Tega doslej še nisem predaval, morda pa je prav, da enkrat začnem. Najbrž že deset let naj bi se datoteke v Pythonu odpirale takole:

```
with open("tabelica-kolesa.txt", "w") as f:
    f.write(f"{'Kolo':6}{{'Število voženj':>19}{{'Razdalja':>19}{{'Višina':>19}}\n")
    f.write("-" * (6 + 3 * 19) + "\n")
    for kolo in voznje:
        voz, raz, vis = voznje[kolo], razdalje[kolo], visine[kolo]
        f.write(f"{'kolo':12}{voz:5} ({voz / sk_voz:>5.1%}) {'raz:10} ({raz / sk_raz:>5.1%}) {\n")
    f.write("-" * (6 + 3 * 19) + "\n")
```

Stavek `with` dela z rečmi, ki obstajajo znotraj nekega bloka (uradno ime za to reč je "kontekst"), ko se izvajanje bloka konča, pa želijo biti o tem obveščene, ker morajo opraviti kakšna "zaključna dela". Datoteka je že taka reč: v začetku bloka jo odpremo, ko je konec bloka, pa je o tem obveščena in se "sama" zapre. Za `as f` pa priredimo spremenljivki `f` rezultat funkcije `open`.

Isto velja za branje: celo `csv.reader` bi morali uradno klicati tako:

```
with open("kolesa.txt") as f:
    for vrstice in csv.reader(f):
        ... in tako naprej
```

Mislite si, kar si hočete. Razumite, če hočete. Datoteke lahko odpirate z običajnim klicem `open` in jih ne zapirate (tako kot smo počeli že od začetka semestra), ali pa jih zapirate s `close`. Kar piše v tem razdelku pa ignorirate. Skoraj gotovo se vam ne bo zgodilo nič. Lahko pa uporabljate `with`.